REMARKS

Responsive to the Written Restriction mailed June 23, 2010, Applicants submit the following amendments and remarks, and request reconsideration and allowance of the application including claims 1-5, 8-24, and 26-29 as set forth herein.

Status of the claims

The Office Action reports examination of claims 1-5 and 8-29.

Claims 1-5 and 8-29 stand rejected under 35 U.S.C. § 102(e) as allegedly anticipated by West et al., U.S. Pub. No. 2002/0140738 A1 (hereinafter "West").

Claims 2 and 3 are objected to for certain cited informalities.

The objections are addressed

The formal amendments to claims 2 and 3 are made in accordance with the Examiner's suggestion. Accordingly, Applicants request reconsideration and withdrawal of the objections to claims 2 and 3.

Other claim amendments

Claim 11 is amended to clarify inserting the control event into the event queue, wherein response of the application program instance to the inserted control event affects execution of the application program instance.

Claim 17 is amended to recite generating and inserting into an event queue at least one initiating event, the at least one initiating event being detected in the event queue and acted upon by the application program or a resource accessed by the application program to affect a user interface of the application program instance.

Claim 24 is amended to recite executing a control program that operates independently of the selected program and that controls the selected program by generating events that are received by the operating system and placed into a program event queue associated with the selected program. **Claim 26** is amended to recite executing a control program independently from the controlled application, wherein the executing control program is configured to control the controlled application by inserting one or more control events into the application event queue. These amendments find support at least in the disclosed control program (20) and general-purpose application program (14) which are independently executing

programs and wherein the control program (20) controls the application program (14) by generating events as recited in claim 24. *See, e.g.* Fig. 1 and related text. **Claims 27 and 29** are amended for consistency with amended claim 26.

The West reference

West discloses an event-driven application program. As shown in West Fig. 2, "the system monitors a current event on its event queue, and determines whether the current event, e.g., a user action or a message sent from another processing device is an application event (step 110)." West ¶[0020]. The application program then takes an action depending upon the type of event. Examples include a create event opening a window for the application program or a destroy event closing said window. *Id.* West also discloses "focus" events which cause the menu system to be directed the application program ("focus gained") or taken away ("focus lost"). West ¶[0021].

West is principally directed to approaches for coordinating menu and toolbars in a Multiple Document Interface (MDI). West ¶[0002]. "In an MDI (Multiple Document Interface) environment, a single parent window contains any number of child windows. The menus and/or toolbars associated with each child window are displayed on a frame of the parent window." *Id.* Toward this end, West discloses the process flow of West Fig. 2, in which responsive to detection of a "create" application event (110) in the event queue (100) an application program container (140) is created (130) and loaded with menu/toolbar properties contained in menu policies or properties (145). More particularly, "The create container function reads one or more policies from a properties file 145, and then integrates the policies into a container for the application (hereinafter an application container 140)." West ¶[0021]. West does not disclose any more detail as to how the application container is accessed by the application. However, it is evident that the West approach requires that the application be pre-configured to be able to read and utilize the information contained in the application container (140). In other words, West is taking advantage of a "built-in" capability of the application to read a certain type of container. If the application does not have this built-in capability, then it cannot operate with the West menu/toolbar coordination system.

Later, if focus on the application program is lost, "the method de-applies the policies for the application that lost the focus (step 160), and a new menu/tool bar is

created in instantiation step 180, wherein a new menu/tool bar 190 is generated based upon the policies of the active applications, taking into account the change in focus." West ¶[0022]. Similar processing occurs in response to detection of a "Destroy" application program event in the queue (100). West ¶[0021]; *see also* Fig. 2 parallel path off of "Destroy" leading from the decision block (120).

Finally, if a lost focus is later regained, "the method applies the policies 155 of the container 140 of the application that gained focus, the container 140 of the application that gained focus is combined with the current container 170, and the method moves to the instantiation step 180, wherein a new menu/tool bar 190 is generated based upon the policies of the active applications, taking into account the change in focus." West ¶[0022].

The claims present patentable subject matter and should be allowed

Claim 11 recites a storage medium encoding instructions which when executed on a computer perform a method for controlling an instance of an event driven application program, the method comprising: monitoring an event queue to detect a selected event associated with the application program instance; and responsive to detecting the selected event, generating a control event and inserting the control event into the event queue, wherein response of the application program instance to the inserted control event affects execution of the application program instance.

West discloses monitoring an event queue (100) (West ¶[0020]) to detect a selected event associated with the application program instance (e.g., a create event or a destroy event, *see* West ¶[0021].

However, West does not fairly suggest, much less expressly or inherently disclose *responding* to the selected event *by generating a control event and inserting the control event into the event queue*, wherein response of the application program instance to the inserted control event affects execution of the application program instance. In contrast, West discloses responding to the detecting of the selected event (110, 200) by performing various operations (130, 155, 160, 180). There is no

suggestion that any of these operations involve "generating a control event and inserting the control event into the event queue".

In this regard, it should be noted that the element (100) of West Fig. 2 is not actually the event queue itself, but rather is a *step* (100). "At step 100, the system monitors a current event on its event queue, and determines whether the current event, e.g., a user action or a message sent from another processing device is an application event (step 110)." Thus, the inclusion in West Fig. 2 of lines feeding into the step (100) do *not* denote inserting anything into the event queue. This is consistent with the text of West, e.g. (referring to the right-most line feeding into step (100): "If the event is neither an application event or a focus event, control is returned to step 100." This does not fairly suggest *responding* to a selected event by *generating a control event and inserting the control event into the event queue.*

The operations of claim 11, namely the monitoring operation and the generating/inserting operation, cooperate to affect execution of the application program instance in response to an occurrence of the selected event. The execution is affected in an efficient way, because the method does *not* need to affirmatively affect the execution of the application program instance. Rather, the method causes the application program instance to affect its *own* execution through the recited operations of detecting to the selected event and responding by generating a control event and inserting the control event into the event queue.

West does not fairly suggest, much less expressly or inherently disclose, this cooperative combination. West discloses monitoring an event queue to detect a selected event associated with an application program instance. But, West responds to the detected event in an entirely different way – it *affirmatively performs* one or more of the various operations (130, 155, 160, 180) (which operations being performed depending upon which event is detected) and generates content, e.g. container (140), that the controlled program is pre-configured to read and interpret. There is no suggestion of causing the application program to perform such operations by generating and inserting a control operation into the event queue.

Claim 17 recites a storage medium storing a control program comprising executable instructions for controlling an application program, the control program performing a method comprising: detecting an initiation of an instance of the

application program; and prior to a user input, generating and inserting into an event queue at least one initiating event, the at least one initiating event being detected in the event queue and acted upon by the application program or a resource accessed by the application program to affect a user interface of the application program instance.

West discloses detecting an initiation of an instance of the application program, i.e. the left branch coming out of decision operation (120) in West Fig. 2. West further discloses *performing* initiating events (130) responsive to the detecting.

However, West does not fairly suggest, much less expressly or inherently disclose, *generating and inserting into an event queue at least one initiating event*, so as to produce the response of the at least one initiating event being detected in the event queue and acted upon by the application program or a resource accessed by the application program to affect a user interface of the application program instance.

Claim 21 recites an apparatus comprising: a computer including (i) an operating system that handles events generated by a user input, by the operating system, or by programs or objects operating under the operating system and (ii) an application program operating under the operating system; and a storage medium encoding instructions which when executed on the computer define a control program operating under the operating system, the control program generating a control event that is detected and acted upon by the application program to cause the application program to perform a selected operation.

West discloses a computer including (i) an operating system that handles events generated by a user input, by the operating system, or by programs or objects operating under the operating system (e.g., West ¶[0020] and step (110)) and (ii) an application program operating under the operating system (e.g., West ¶[0021]). West further discloses (or at least fairly suggests) a storage medium encoding instructions which when executed on the computer define a control program operating under the operating system. *See, e.g.* West ¶[0051] (describing computer readable media); West Fig. 2 (depicting an application program menu control program).

However, West does not fairly suggest, much less expressly or inherently disclose, the control program generating a control event that is detected and acted upon by the application program to cause the application program to perform a selected operation. Rather, the control operations of West Fig. 2, even if broadly

construed as "control events", are not detected and acted upon by the application program to cause the application program to perform a selected operation, but rather are affirmatively executed, and *not* by the application program.

Claim 24 recites a storage medium encoding instructions which when executed on a computer in conjunction with concurrent execution of an operating system and a selected program perform a method comprising: executing a control program that operates independently of the selected program and that controls the selected program by generating events that are received by the operating system and placed into a program event queue associated with the selected program, the selected program performing predetermined operations in response to the generated events, the predetermined operations producing a desired modification in an execution of the selected program.

West does not disclose such a control program. West does disclose a control program, but *not* a control program that controls the selected program *by generating events that are received by the operating system and placed into a program event queue associated with the selected program*. Rather, West's control program affirmatively performs control operations including reading the properties file (145) and generating the component container (140) which the application program is pre-configured to read.

Moreover, West does not disclose or fairly suggest a control program *that operates independently of the selected program*. Rather, West's control program is an integral part of the controlled application. West's control program relies upon the controlled application being pre-configured to read the component container generated by West's control program. In effect, West's control program is a module or resouorce of the controlled application program, namely a module which generates the menu system of the controlled program. The usefulness of such an approach is restricted to those applications that are pre-configured to read and interpret content as generated by the West program. If the application is not thusly pre-configured, it cannot be controlled by the West approach. In contrast, the control program of claim 24 can be used to control any event-driven program.

Claim 26 recites a storage medium encoding instructions which when executed on a computer perform a method for controlling execution of a controlled application operating in an operating system environment loaded onto the computer that maintains an application event queue associated with the controlled application, the method comprising executing a control program independently from the controlled application, wherein the executing control program is configured to control the controlled application by inserting one or more control events into the application event queue, the one or more control events producing a predetermined response by the controlled application or a resource accessed by the controlled application.

West does not fairly suggest, much less expressly or inherently disclose, a control program configured to control a controlled application *by inserting one or more control events into the application event queue*, the one or more control events producing a predetermined response by the controlled application or a resource accessed by the controlled application. Rather, West's approach is to affirmatively perform control operations.

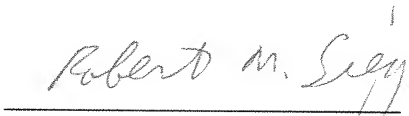
Moreover, West's control program does not execute independently from the controlled application. Rather, West's control program is an integral part of the controlled application, in effect a module of the controlled application that generates content that the controlled application is pre-configured to utilize.

CONCLUSION

In view of the foregoing, Applicants respectfully submit that the application including all claims 1-5, 8-24, and 26-29 as set forth herein present set forth patentable subject matter, meet all statutory requirements, and should be allowed. Accordingly, Applicants respectfully request allowance of the application including all claims set forth herein.

Respectfully submitted,

FAY, SHARPE LLP



Thomas E. Kocovsky, Jr.
Reg. No. 28,383
Robert M. Sieg
Reg. No. 54,446
The Halle Building, 5th Floor
1228 Euclid Avenue
Cleveland, OH 44115
Phone: (216) 363-9000
Fax: (216) 363-9001

N:\MDTI\200001\MDTI200001.AMD.B.DOC